Optimisation du Probième d'Ordonnancement à Machines Parallèles dans Hadoop

Aymen Jlassi, Patrick Martineau, Vincent Tkindt

Optimisation du Problème d'Ordonnancement à Machines Parallèles dans Hadoop

Aymen Jlassi[1,2], Patrick Martineau[1], Vincent Tkindt[1]

[1] Université François-Rabelais de Tours,
CNRS, LI EA 6300, OC ERL CNRS 6305, Tours, France
{patrick.martineau,tkindt}@univ-tours.fr
[2] Groupe Cyrès {aymen.jlassi}@etu.univ-tours.fr

Mots-clés : *Ordonnancement à machines parallèles, Optimisation, Heuristique.*

1 Introduction

On s'intéresse dans ce travail à l'amélioration du fonctionnement d'un logiciel de traitement distribué de gros volumes de données nommé Hadoop. Notre objective est l'optimisation de l'ordonnancement d'un ensemble de travaux sur une architecture à machines parallèles, en se restreignant aux travaux du type Map / Reduce. Puisque le problème est NP-difficile et les instances considérées sont de grande taille, on propose deux heuristiques de résolution basées sur des algorithmes de listes.

Le modèle Map / Reduce est un modèle de développement introduit par Google en 2004 dont l'objectif est de faciliter le développement et l'exécution d'applications parallèles. Ce modèle impose un cadre de développement : (1) Un travail Map / Reduce est composé de deux types de tâches : les tâches Map et les tâches Reduce. Les tâches Map s'exécutent avant les tâches Reduce. Les tâches Map s'occupent d'effectuer les calculs alors que les tâches Reduce les agrègent. Implicitement, les données de sortie des tâches Map sont transférées à travers le réseau aux machines exécutant les tâches Reduce associées. (2) Les tâches Map et les tâches Reduce ne supportent pas d'être interrompues : si une tâche est interrompue, elle sera relancée comme si elle s'exécutait pour la première fois.

2 Contexte du travail

On considère un ensemble de machines, chacune est caractérisée par (1) un ensemble de slots qui exécutent des tâches Map et un ensemble de slots qui exécutent des tâches Reduce, (2) une capacité mémoire (RAM) et une quantité de disque dur (3) une fréquence de calcul de chaque slot. L'utilisation des ressources de chaque machine induit à un coût (coûts d'utilisation de la RAM, du disque dur et des slots). Chacun des travaux est décomposé en un ensemble de tâches du type Map et un ensemble de tâches du type Reduce. Lorsqu'une tâche s'exécute, elle nécessite une quantité de RAM, de slot, de disque dur et permet de traiter un ensemble de blocs de données situés sur le système de fichiers. Afin d'assurer la communication entre deux tâches de types différents, une bande passante de communication doit être réservée. Tous les blocs de données ont la même taille, chacun est caractérisé par un nombre de réplications, la localisation des réplications et la bande passante nécessaire pour migrer un bloc d'une machine à une autre. Dans ce travail, on traite la minimisation de la durée totale d'exécution des tâches.

2.1 Travaux précédents

Dans un précédent travail [1], nous avons modélisé de façon mathématique le problème d'ordonnancement dans Hadoop comme un problème d'ordonnancement à machines parallèles. Nous nous sommes focalisés sur l'étude du problème hors ligne avec des tâches de durée égale à 1 et le C_{max} comme critère. Le modèle obtenu a été évalué sur des instances de petites tailles. Le solveur CPLEX ayant résolu des instances jusqu'à :

- 46 tâches (dont 33 tâches map) sur deux machines (9 slots map et 5 slots reduce)
- 30 tâches (dont 18 tâches map) sur trois machines (11 slots map et 5 slots reduce)

Cette première étude nous a permis de mettre au point des bornes de référence pour le présent travail.

2.2 Problématiques abordées

Dans ce travail, nous considérons des tâches de durée unitaire et nous proposons dans une première partie une solution basée sur des algorithmes de listes pour le mode hors ligne. Les tâches de type map sont ordonnancées par priorité sur les slots des machines contenant les données qu'ils traitent afin d'éviter les pénalités dues à leur transfert. Si, à une date précise, on ne peut pas respecter cette contrainte, les tâches sont ordonnancées sur les slots les moins chargés. Dès qu'un ensemble de tâches map d'un travail particulier est terminé, les tâches reduce associées sont ordonnancées suivant la règle FIFO sur les slots les moins chargés. Les résultats obtenus sont proches des résultats obtenus par la résolution du modèle mathématique.
Dans une deuxième partie, nous nous intéressons au problème en-ligne. On propose une version adaptée de cette heuristique. On adapte la méthode proposée par Shmoys, Wein et Williamson [3] pour assurer le passage d'un algorithme hors ligne à un algorithme en ligne. Cette heuristique est basée sur SPT (Shortest Processing Time) et tient compte, elle aussi, de la localité des exécutions par rapport aux données.
Nos expérimentations ont porté sur des instances de grande taille, pour lesquelles on compare nos résultats à ceux obtenus avec des heuristiques de la littérature telle que l'heuristique Fair [2]. Les critères mesurés dans les simulations sont le coût d"utilisation des ressources et la consommation réseau.
Les heuristiques et les résultats proposés seront présentés durant la conférence.

3 Perspectives

Dans ce travail, nous avons considéré à minimiser la durée d'achèvement C_{max}. La prochaine étape de notre travail consiste à minimiser, en plus, la consommation énergétique qui représente des coûts d'utilisation de la grappe. L'étude est en conséquence basée sur une approche d'optimisation bicritère.

Références

[1] Aymen Jlassi, Patrick Martineau, Vincent Tkindt. *Modélisation Mathématique du problème d'Ordonnancement dans Hadoop* ROADEF - 15ème congrès annuel de la Société française de recherche opérationnelle et d'aide à la décision, Feb 2014, Bordeaux, France

[2] Yongcai Tao and Qing Zhang and Lei Shi and Pinhua Chen. *Job Scheduling Optimization for Multi-user Map / Reduce Clusters.* Parallel Architectures, Algorithms and Programming (PAAP), 2011 Fourth International Symposium on, vol., no., pp.213-217, 9-11 Dec. 2011.

[3] Shmoys, David B. and Wein, J. and Williamson, D.P. ; *Scheduling parallel machines online.* Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium, pp.131-140, 1-4 Oct 1991.